1. 单例模式（Singleton）
特点：

唯一性：确保一个类只有一个实例。
全局访问点：提供一个全局的访问点来访问该实例。
用途：常用于需要全局唯一对象的场景，例如日志记录器、配置管理器、数据库连接等。
应用场景：

配置管理：一个应用通常只需要一个配置管理器，单例模式确保配置管理器实例唯一。
日志系统：日志记录器需要在整个应用中唯一，单例模式可以保证这一点。
数据库连接池：为了管理数据库连接，一个应用通常需要一个唯一的连接池管理器。
2. 工厂模式（Factory）
特点：

创建对象：定义一个用于创建对象的接口，让子类决定实例化哪个类。
解耦：将对象的创建和使用解耦，使用者无需知道具体的创建逻辑。
扩展性：易于扩展，增加新产品无需修改现有代码。
应用场景：

产品创建：如在图形界面库中，不同的按钮、窗口等控件可以通过工厂模式来创建。
数据库访问：根据不同的数据库类型（如 MySQL、PostgreSQL），工厂模式可以动态创建相应的数据库访问对象。
日志框架：可以根据不同的日志记录方式（如文件、控制台、网络）创建相应的日志记录对象。
3. 观察者模式（Observer）
特点：

一对多依赖：定义对象间的一对多依赖关系，当一个对象状态改变时，所有依赖的对象都会得到通知并自动更新。
松耦合：观察者和被观察者之间松耦合，增加观察者不会影响被观察者的实现。
动态订阅：观察者可以在运行时动态订阅或取消订阅。
应用场景：

事件系统：如 GUI 事件处理，用户操作（点击、输入等）会触发相应的事件处理。
数据变化通知：如在 MVC 模式中，模型数据变化通知视图更新。
实时数据更新：如股票交易系统中，股价变化通知订阅者更新显示。
4. 装饰模式（Decorator）
特点：

动态扩展：在不改变对象接口的情况下，动态地给对象添加职责。
透明性：装饰模式对使用者透明，装饰后的对象仍然可以像原对象一样使用。
可组合性：多个装饰器可以组合使用，叠加对象的职责。
应用场景：

图形界面：如窗口组件可以动态添加滚动条、边框等装饰。
日志功能：在不修改已有代码的情况下，动态添加日志记录功能。
数据流处理：如在输入输出流处理中，可以动态添加缓冲、加密、压缩等功能。
总结
这些设计模式在软件开发中提供了解决特定问题的有效方法：

单例模式：适用于需要全局唯一实例的场景。
工厂模式：用于对象创建的场景，特别是当对象创建逻辑复杂或需要动态决定具体类型时。
观察者模式：适用于事件驱动系统和需要通知机制的场景。
装饰模式：适用于需要动态扩展对象功能而不改变其接口的场景。
将这些模式及其应用场景记录在团队的协作开发平台上，可以帮助团队成员理解和使用这些模式，从而提高项目的设计质量和代码的可维护性。